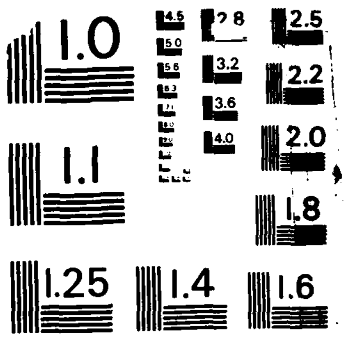
AD-A182 330 SYNTACTIC PATTERN RECOGNITION FOR RADAR TARGET 1/1
IDENTIFICATION(U) OHIO STATE UNIV COLUMBUS
ELECTROSCIENCE LAB O S SANDS ET AL MAY 87
UNCLASSIFIED ESL-718048-2 N00014-86-K-0202 F/G 17/9 NL

END
8-87
DTIC


MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

(12)

DTIC FILE COPY

T · H · E
OHIO
STATE
ERSITY

SYNTACTIC PATTERN RECOGNITION FOR RADAR TARGET IDENTIFICATION

by

O.S. Sands and F.D. Garber

The Ohio State University

ElectroScience Laboratory

Department of Electrical Engineering
Columbus, Ohio 43212

Technical Report 718048-2
Contract N00014-86-K-0202
May 1987

Department of the Navy
Office of Naval Research
800 N. Quincy St.
Arlington, Virginia 22217

AD-A182330

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO.	2.	3. Recipient's Accession No.
4. Title and Subtitle Syntactic Pattern Recognition for Radar Target Identification			5. Report Date May 1987
			6.
7. Author(s) O.S. Sands and F.D. Garber			8. Performing Organization Rept. No 718048-2
9. Performing Organization Name and Address The Ohio State University ElectroScience Laboratory 1320 Kinnear Road Columbus, Ohio 43212			10. Project/Task/Work Unit No.
			11. Contract(C) or Grant(G) No (C) N00014-86-K-0202 (G)
12. Sponsoring Organization Name and Address Department of the Navy, Office of Naval Research 800 N. Quincy Street Arlington, VA 22217			13. Type of Report & Period Covered Technical
			14.

15. Supplementary Notes

16. Abstract (Limit: 200 words)

A syntactic pattern recognition system is proposed for use in radar target identification. The system utilizes one of three different level-crossing based pattern representation schemes. Likelihood functions are estimated from relative frequency densities and are used for classification of the symbolic pattern representation. Grammatical inference is used to derive a syntax analysis algorithm from the likelihood function classifier. Classifier performance is estimated using Monte-Carlo simulations. Directions of further research on this promising topic are discussed.

17. Document Analysis a. Descriptors

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement Approved for public release, distribution is unlimited.	19. Security Class (This Report) Unclassified	21. No of Pages 72
	20. Security Class (This Page) Unlimited	22. Price

(See ANSI-Z39.18) See Instructions on Reverse OPTIONAL FORM 272 (4-77) (Formerly NTIS-35) Department of Commerce

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

1.1 THE RTI PROBLEM

Identification of unknown targets using information contained in radar returns is the subject of this study. The identification is defined in a 1 of N sense where a description is available of each of the N choices. The envisioned radar system is a general purpose, multifrequency, multipolarization system. Specifically, it operates in the range from 8 to 58 MHz in horizontal transmit, horizontal receive mode (see [1].) These frequencies represent the *resonant region* of a catalog of radar targets which are used for the experimental phases of the study (see [2].) That is, the band of frequencies with wavelengths which are approximately equal to the dimensions of the target.

The data from the receiver of the radar system is a set of complex numbers, one number for each frequency in the operational range of the radar. This vector of complex numbers represents the *in-phase* and *quadrature* components of a coherent continuous wave radar system as a function of frequency (see [3]). For experimentation purposes, simulated radar returns were obtained from the compact radar range as discussed in [4]. The compact range data has been normalized so that all system related parameters have been removed from the measurements. The compact range data is in units of dBm^2 which is the radar cross section of the target, normalized to 1 square meter. This unit of measurement is also used for noise superimposed on the data.

Traditionally, classification methods based upon decision-theoretic concepts have been applied to the radar target identification problem. The historic popularity of decision theoretic applications is primarily due to the well-defined sense of optimality for these schemes. In contrast with the more traditional decision theoretic classification methods, this study investigates a classifier based upon syntactic pattern recognition principles. This is not to say that decision theoretic principles are abandoned entirely, rather they are used to help bring a sense of optimality to a syntactic classifier.

1.2 SYNTACTIC SYSTEMS

A classifier, based on the syntactic approach to pattern recognition, classifies measured patterns by means of a *structural description* of the pattern measurement [5, Chapter 1]. This is in contrast to the feature-based description of target measurements employed by decision-theoretic methods.

A syntactic system consists of a pattern representation section, which processes the pattern measurements into a structural description of the target, and

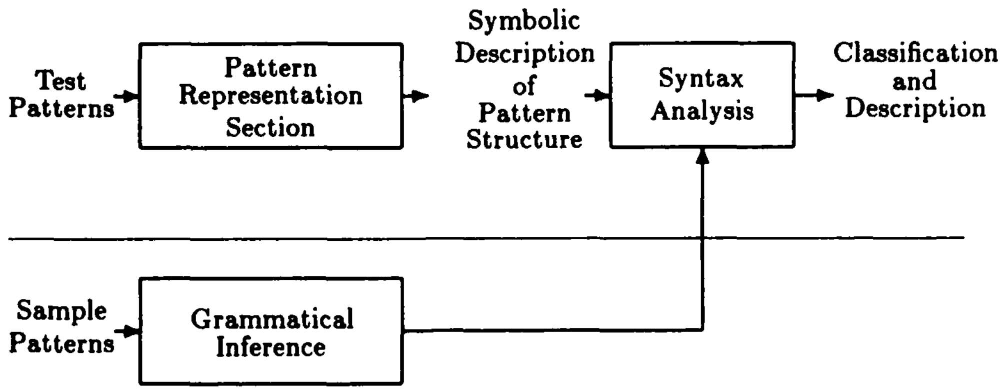

Test Patterns
Pattern Representation Section
Symbolic Description of Pattern Structure
Syntax Analysis
Classification and Description
Sample Patterns
Grammatical Inference

Figure 1: Syntactic Pattern Recognition System

a syntax analysis section, which uses the structural description to deduce the identity of the unknown target (see figure 1.) A grammatical inference section defines the syntax analysis section, usually from a set of sample patterns which have a known classification and description. The grammatical inference section is executed during a learning session, which occurs prior to classification. The grammatical inference section may be seperate from the physical system which performs the classification. However, it is included in the definition of a syntactic system as a whole.

For purposes of radar target identification, the process of generating a structural description consists of converting localized sections of a radar cross-section measurement as a function of frequency (pattern) to a set of discrete symbols called *primitives*. The resulting primitives are then linked together to form a *symbolic* *description* of the structure of the sampled frequency spectra. The linking of the primitives may take the form of a directed graph such as a tree, or can be as simple as concatenation of the primitives. The exact form of the primitives, as well as linking information, depends upon the definition of the pattern representation scheme.

Language-theoretic methods are generally used for syntax analysis in syntactic systems. In syntactic pattern recognition systems, an analogy is drawn between the structure of the symbolic target description and the *syntax* of the inferred grammatical system. Thus the name "syntactic" systems. Furthermore, the use of language theoretic methods requires a discrete-symbolic nature of the structural description of the pattern be available. The more general interpretation of the definition of syntactic systems (the one which has been presented in previous paragraphs) considers *syntactic systems* to be *structural systems*. In this interpretation, any system which classifies a pattern from it's structure is included, rather than restricting the class to systems which use language-theoretic classifiers.

The feature of providing a structural description of the measurement pattern may prove to be of significant utility in the radar target identification problem. This is especially significant when the number of classes is large, or when the number of target and catalog measurements is such that the classification task becomes impractical to implement. In addition, it is felt that a *structural* description may provide useful information about the target; even in cases when the system is unable to classify the target as a member of one of the classes in the available catalog.

Syntactic systems implemented for radar target identification should also be considerably less complex than their decision-theoretic counterparts. For example, much less precision in the radar measurements may be required since only the "structure" of the sampled frequency spectra is important. The language-theoretic systems used by syntax analysis sections are the most basic models of computing machines. Implementation of the syntax analysis section by digital computers is therefore, straightforward.

1.3 PURPOSE OF THE STUDY

Determining the feasibility of application of syntactic methods to the radar target identification problem is the focus of Chapter 2. To do this, the symbolic pattern description from a practical pattern representation section is considered to be the observation to a decision-theoretic system. The target is classified from the symbolic pattern representation using optimal decision rules. Classification results of such an optimal system indicate the degree of usefulness of such a pattern representation scheme. From this perspective it is determined if it is *possible* to classify on the basis of the information contained in the symbolic description.

Once feasibility has been established, a language-theoretic syntax analysis algorithm is derived using the decision rules of the decision theoretic system of

Chapter 2 as a basis for grammatical inference. This is the topic of Chapter 3.

CHAPTER II

FEASIBILITY STUDY

2.1 INTRODUCTION

2.1.1 FEASIBILITY OF SYNTACTIC PATTERN RECOGNITION USING PRACTICAL PATTERN REPRESENTATION SCHEMES

It is clear, from an information-theoretic standpoint, that the development of a pattern representation scheme is one of the most crucial parts of the specification of a syntactic radar target identification system. The representation scheme must preserve enough information to distinguish a target, represented by its radar return, from other targets in the given catalog. That is, the statistic formed by the pattern representation section must contain information sufficient to identify the target.

Theoretically, it is always possible to find a pattern representation scheme which produces a symbolic description that is a sufficient statistic. Indeed, such a system could be made by combining a pattern representation scheme that is a decision-theoretic classification system with a syntactic classifier that is merely an identity mapping (i.e., simply announce the conclusion of the decision theoretic system). A syntactic system utilizing such a pattern representation scheme requires all the resources of the decision theoretic system which defines the pattern representation section and offers no advantage over the decision theoretic system. Therfore, such a pattern representation section is considered impractical for a syntactic system. A *practical* pattern representation scheme performs its task using less resources than a decision theoretic system that performs the entire classification process as well as preserving enough information to reliably classify the target. This portion of the investigation demonstrates that a practical pattern representation scheme can preserve enough information to classify an aircraft radar target.

This task is carried out with the aid of a simulation of the envisioned syntactic radar target identification system. For this simulation, radar range data is used to create test patterns for a Monte-Carlo simulation of the complete system as well as formulating the syntax analysis.

2.1.2 SYSTEM AND SIMULATION OVERVIEW

The different pattern representation schemes were restricted to the production of strings for this study. This was done to simplify the simulation and any resulting language-theoretic syntax analysis. The organization of these symbols into strings, is *not* necessary for implementation of a syntactic classifier. Grammatical systems do exist which produce higher dimensional sentences. Indeed, previous investigations of symbol assignments for measurement waveforms indicate that higher dimensional pattern representations would allow more reliable classification [6]. This is probably due to the fact that more complex structural information could be conveyed with such data structures.

The syntax analysis algorithm of the proposed system is initially limited to the classification function. This is implemented with likelihood function tests. The likelihood functions for these tests are empirically derived by the simulation itself. The process of deriving these functions can be thought of as grammatical inference for the system. Likelihood function tests are chosen because of their well defined optimality. Maximum likelihood, maximum a-posteriori and Bayes decision rules can all be implemented as likelihood function tests. The decision rules are optimum in their respective senses when the output of the pattern representation section is considered the observation of a decision-theoretic system.

The performance of the total system consisting of a pattern representation section coupled with a decision-theoretic classifier is evaluated using Monte-Carlo testing. Radar data is simulated by vector addition of radar range measurements and a vector deviate with Gaussian statistics. Confusion matrices, which give average cross-classification for each of the catalog element combinations, as well as average misclassification percentages are computed. Each noise level chosen for testing uses a statistically sufficient number of trials.

2.1.3 SUFFICIENCY EVALUATION

Analytic methods of evaluating the sufficiency of the statistic formed by the pattern representation section could be cumbersome for the various pattern representation schemes. If the classification ability of a system which uses the output of the pattern representation scheme as an observation is adequate then the statistic formed by that pattern representation scheme is assumed to be "sufficient" to some degree. Monte-Carlo testing is used for evaluating the classification ability of the system.

2.2 SYNTACTIC PATTERN RECOGNITION SYSTEM SIMULATION

2.2.1 PATTERN REPRESENTATION SCHEMES

In order to gain insight into the effects of the pattern representation on the performance of syntactic target identification, the performance of systems using three different pattern representation schemes is evaluated. Each of these represents a radar return measurement in terms of "level crossings". This type of representation was, in part, suggested by the results reported in [7] and [8] which indicated that much of the information contained in a waveform is also contained in its zero crossings. Moreover, pattern representations of the level-crossing type realize other properties that are intuitively desirable. For example, in order to reduce the complexity of the syntax analysis section of the classifier, the set of primitives (or symbols) necessary to represent a given pattern is small. In addition, the level crossing type pattern representation sections are easy to implement because of the string nature of their output. Also, the resulting representations are relatively immune to the effects of noise.

SINGLE LEVEL CROSSING

The single level crossing pattern representation scheme forms its output string based on the number of consecutive radar return measurements that lie above or below a pre-determined threshold. After threshold calculation, formation of primitives begins by determining the number of consecutive measurements (starting with the first measurement) which have magnitude *below* the threshold. In case the first measurement lies above the threshold, the first primitive is taken to be zero. The second primitive is the number of consecutive measurements, subsequent to the initial set, that have magnitude *above* the threshold; and so on until all the measurements in the set have been accounted for. Note that the only place a zero can occur is in the first position of the string of primitives.

The pre-determined threshold for this scheme is taken to be the average value of the magnitude of the measurement data. Thus, the threshold is between the minimum and maximum values of magnitudes of the measured sampled frequency spectra. Also notice that the size of the set of primitives for this scheme is not fixed and varies with the number of measurements taken by the system. An example of the single level crossing processing technique is shown in Figure 2 below. This

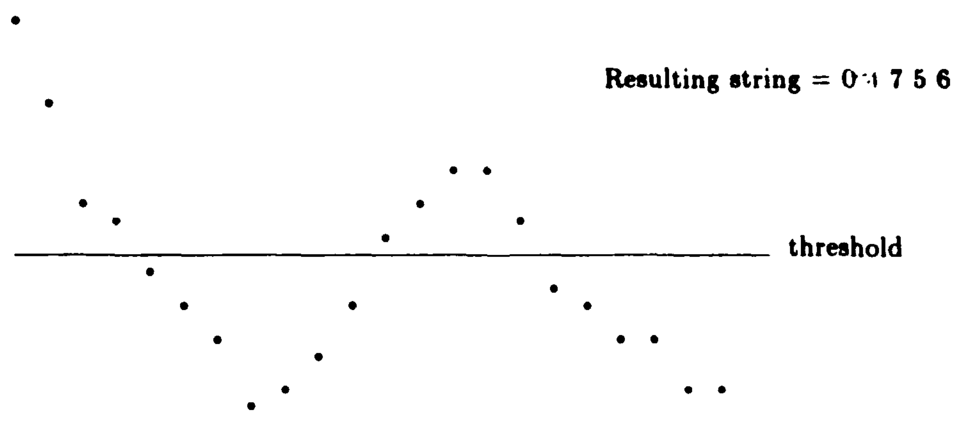


Figure 2: Single level crossing example

scheme is exactly equivalent to producing a vector of n (where n is the number of measurements made) 1s and 0s which would represent *above* and *below* the average value threshold. For this reason it is apparent that the number of possible strings which could be generated by such a system is 2^n. Finite state automata which must accept strings with repeated primitives can contain excessive numbers of states. Clearly, the pattern representation scheme, as presented, produces fewer repeated symbols than the 1/0 vector scheme. Since finite state automata are used for syntax analysis in the second part of the study, this single level crossing scheme is preferable to the 1/0 scheme since fewer states are needed in the generated finite state automata.

OCTANT CROSSING WITH REDUNDANCY REMOVAL

The octant crossing method of pattern representation incorporates the simplicity of level crossing determination into a scheme that also employs partial phase information. The representation algorithm begins by calculating the average value of the magnitude of the measurement data to determine the threshold level as above. During operation, the radar receiver decides whether the measurement has magnitude above or below this threshold, and, in addition, the "relative phase quadrant" in which the measurement lies is calculated. The resulting octant of a radar measurement for each combination of level and quadrant is defined by Figure 3. The categories "above" and "below" refer only to the magnitude of the measurement while the phase ranges refer only to the angle. Finally, redundant (repeated) primitives appearing in a representation string are removed so that any given symbol appears only once in succession. This is refered to as "squeezing"

In the octant representation scheme, the number of possible strings which could be produced is $\frac{4}{3}(7^n - 1)$ where n is the number of measurements made. This is significantly more than the single level crossing algorithm. However, the addition of relative phase information should produce better classification results. Moreover, the implementation of this processing technique is a straightforward extension of the single level crossing algorithm. The squeezing of the strings was used to eliminate repeated symbols within the strings. This simplifies the eventual

octant	phase	magnitude
a	0-90	below
b	90-180	below
c	180-270	below
d	270-0	below
e	0-90	above
f	90-180	above
g	180-270	above
h	270-360	above

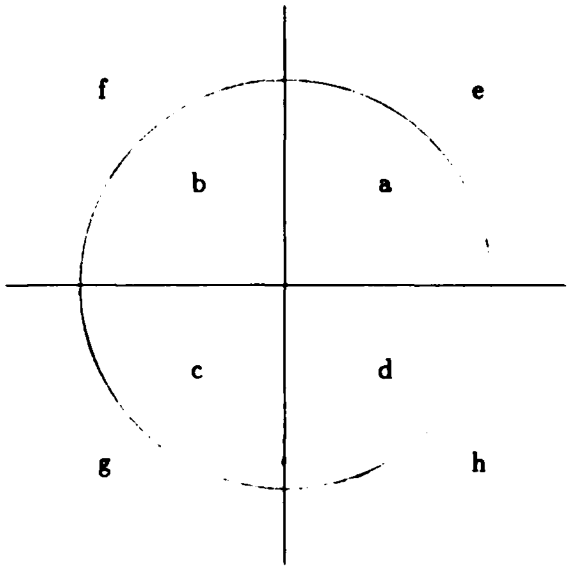


Figure 3: Octant crossing primitive assignments

syntax analysis, as in the single level crossing case.

DOUBLE LEVEL CROSSING WITH REDUNDANCY REMOVAL

The double level crossing method of pattern representation begins by determining the maximum and minimum magnitudes of the radar measurements of interest. Upper and lower thresholds are then chosen so as to divide the measurement range into thirds. During operation, primitives are assigned to the observed measurements according to their position relative to these two thresholds as shown in Table 1. Finally, redundant symbols are removed from the pattern representation string by deleting primitives that are repeated.

The number of possible strings which can be produced by the double level crossing scheme is $3(2^n - 1)$, where n is the number of measurements made. Actual

Table 1: Double level crossing primitive assignments

level	magnitude
a	below lower threshold
b	above lower threshold, below upper threshold
c	above upper threshold

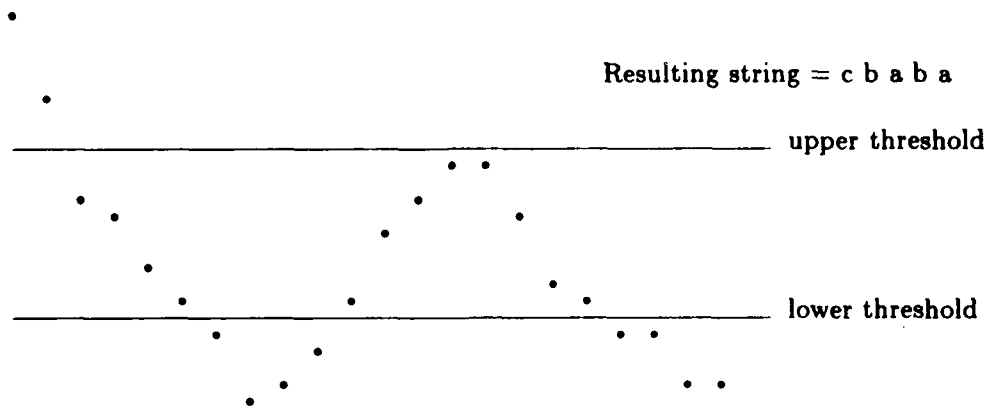


Figure 4: Double level crossing example

implementation of this technique requires little more complexity than the single level crossing scheme. An example of the double level crossing technique is shown in Figure 4. Squeezing was used for syntax analysis simplification as before.

PATTERN REPRESENTATION SUMMARY

It is expected that the single level crossing scheme may be least affected by noise and interference. This is because the number of possible strings produced by the single level crossing representation scheme is smaller than any of the other schemes (see Table 2.) The octant crossing scheme is expected to produce sig-

Table 2: Number of strings generated by n measurements

# of	Pattern rep. scheme		
Measurements	SLC	Octant	DLC
n	2^n	$\frac{4}{3}(7^n - 1)$	$3(2^n - 1)$
6	64	156864	189
11	2048	2.636×10^9	6141

nificantly better results than the other schemes since it includes relative phase information. While more strings can be generated by double level crossing than single level crossing, it is left to the testing phase to determine if the scheme is able to take advantage of the increased potential for information exchange.

2.2.2 SYNTAX ANALYSIS

The syntax analysis for this study is limited to classification using likelihood function tests. The likelihood functions for this system use strings as the independent variable. Since any ordering placed on these strings would be artificial, explicit versions of the likelihood functions are required to form the likelihood function tests. It may be possible to analytically derive likelihood functions for a particular pattern representation scheme, however, the complex nature of the schemes generally prohibit analytic methods. Furthermore, any analytic methods valid for one pattern representation scheme and noise model would not, necessarily, be valid for a different noise model or pattern representation scheme. It is for these reasons that the densities which make up the likelihood functions are approximated with relative frequency densities, or histograms. This method is valid for any noise model or pattern representation combination as long as they are both, well defined. Likelihood functions are formed from the individual densities by indexing the individual densities with their appropriate catalog element. To form the densities, sets of radar measurements are simulated. The densities are formed by repeating the simulation process of the radar measurements and recording the number of times that each resulting string appears. The number of trials used for this process is statistically sufficient for a desired level of accuracy. This fact is demonstrated below.

The optimality of the likelihood function tests is compromised by estimation. To what extent the resulting decision rules become sub-optimal remains unknown and this question is not fully addressed in this study. A further consequence of this estimation procedure is the possibility that a string which did not occur during likelihood function estimation may occur during Monte-Carlo testing. When this occurs, the conditional probability of this string is taken to be zero for all catalog elements and any decision rule which uses these functions is not able to classify such a string. This suggests that statistics should be compiled, during Monte-Carlo testing, of the number of times that no decision can be made to give an estimate of the percentage of *unknown-classification.* Decision rules could be extended to simply guess which of the catalog elements is occuring in such cases. However, the inability to classify is information in and of itself and the appropriateness of using a randomized decision rule should be carefully examined.

A theoretical lower bound on the accuracy of the likelihood function estimate is now given. This lower bound is expressed by the probability that our estimate of the actual value of the density is in error by *more* than some ϵ is *less* than some δ or:

$$\mathbf{Pr}\{|\hat{P}(h_i) - P(h_i)| \geq \epsilon \} \leq \delta \tag{2.1}$$

Where $\hat{P}(h_i)$ is the estimate of the value of the density, $P(h_i)$ is the actual value of the density, ϵ is the estimation error, and δ is the probability that the estimate differs from the true value by more than ϵ. By both Chebyschev's inequality and the central limit theorem the number of samples needed to meet this condition is

$$n = \frac{k}{\epsilon^2} \tag{2.2}$$

where k is some constant which depends on δ, the possible values which the true value of the density can take on.

The simulation, however, uses an adaptive scheme to insure that the density estimates have converged to the true densities within an allowable error. A fixed number(50) of simulated sample patterns are generated and an estimate of the true density is formed with a relative frequency density. A second density is formed by adding more (50) sample patterns and the two densities are compared. If there are no added categories (new strings) and the elementwise difference between the two densities is sufficiently small then the two densities are said to be equal and the algorithm is said to converge. Otherwise, the process continues until a suitable estimate for the true density is found.

For simulation purposes, the likelihood functions are represented in tabular form. This may be undesirable for a practical system because of the tremendous memory requirements and because of the time which would be required for searching the list. The simulation uses hashing functions based on the length of the string to speed execution. Practical systems which would use likelihood functions would likely implement the functions with a set of stochastic automata (see Chapter III. This could be done with the grammatical inference techniques due to Bierman and Feldman, given in [5].

An example of an estimated likelihood function is now given. The densities which make up the function were made using a noise level of 5 dBm^2 training noise, superimposed on the radar return data from 5 different aircraft targets at 0° azimuth, 0° elevation aspect. The single level crossing pattern representation was used and 6 frequencies were used from 8 to 58 MHz. by 5 MHz. This number of frequencies was suggested by a feature extraction study performed on the data from the compact range (see [9].) Table 3 gives data regarding the creation of the likelihood function. As an example, consider the entry corresponding to target number 2. The density created for this target contains probabilities for 21 strings,

Table 3: Likelihood function statistics

catalog no.	no. of elements	no. of samples	CF by Pr Df.	CF by A. S.
1	12	301	1	4
2	21	1201	14	9
3	23	351	1	5
4	1	151	1	1
5	2	1101	20	1

1201 samples were needed for convergence. Convergence of the density failed because samples were added 9 times (CF by A. S.). Convergence failed because the elementwise difference of the probabilities was too great for at least one element 14 times (CF by Pr. Df.).

The actual likelihood function is given in Table 4. The individual densities are given in sequential order starting with catalog element 1. For each entry in the tabular densities, the length of each entry is given, followed by its corresponding probability estimate. The number of times that the entry occured is listed under *accum#* (the strings accumulator) and the actual string-entry is listed under *string*.

It is assumed that the only consequence of the probability difference criterion for density convergence is that the probability estimates of the strings which are non-zero are in error by no more than the amount which is used for comparison of the elementwise difference in the probability estimates. For the experiments in this report that number was set to .01.

It is also assumed that the two criteria operate independently. Since the effect of a convergence failure is to add more samples, a convergence failure from one of

Table 4: Likelihood function example

Catalog element 1			
length	prob	accum#	string
9	0.00332	1	0 1 1 2 1 1 1 1 3
7	0.02990	9	0 1 1 4 1 1 3
7	0.00997	3	0 1 1 2 1 1 5
7	0.00664	2	0 1 1 2 1 3 3
7	0.04319	13	0 1 1 2 1 2 4
7	0.00997	3	0 1 1 5 1 1 2
7	0.00332	1	0 1 1 4 2 1 2
5	0.60133	181	0 1 1 5 4
5	0.17608	53	0 1 1 4 5
5	0.03654	11	0 1 2 4 4
5	0.06977	21	0 1 1 6 3
5	0.00997	3	0 1 2 3 5

Catalog element 2			
length	prob	accum#	string
9	0.03664	44	1 1 1 1 3 1 1 1 1
9	0.00250	3	1 1 1 2 2 1 1 1 1
8	0.47627	572	1 1 1 1 3 1 1 2
8	0.17069	205	1 1 1 1 3 1 2 1
8	0.01499	18	1 1 1 2 2 1 1 2
8	0.01499	18	1 1 1 2 2 1 2 1
7	0.00666	8	1 1 1 1 3 1 3
7	0.01082	13	1 3 3 1 1 1 1
7	0.00250	3	1 4 2 1 1 1 1
7	0.00167	2	1 1 1 1 5 1 1
6	0.13988	168	1 3 3 1 1 2
6	0.09159	110	1 3 3 1 2 1
6	0.00999	12	1 1 1 1 5 2
6	0.00333	4	1 1 1 1 6 1
6	0.00583	7	1 4 2 1 1 2
6	0.00083	1	1 1 1 2 5 1
6	0.00333	4	1 4 2 1 2 1
5	0.00083	1	1 3 3 1 3
4	0.00416	5	1 3 5 2
4	0.00167	2	1 3 6 1
4	0.00083	1	1 4 5 1

Table 4: (continued) Likelihood function example

Catalog element 3			
length	prob	accum#	string
9	0.00570	2	0 1 1 3 1 1 2 1 1
7	0.04843	17	0 1 1 3 1 1 4
7	0.04274	15	0 1 1 3 4 1 1
7	0.05698	20	0 1 1 4 3 1 1
7	0.00570	2	0 1 1 4 1 1 3
7	0.00285	1	0 1 1 1 1 2 5
6	0.00285	1	0 5 2 1 2 1
5	0.25641	90	0 1 1 4 5
5	0.00285	1	0 7 2 1 1
5	0.21937	77	0 1 1 3 6
5	0.00570	2	0 6 3 1 1
5	0.00570	2	2 3 4 1 1
5	0.01709	6	0 5 1 1 4
5	0.02279	8	0 1 1 5 4
5	0.01709	6	0 5 4 1 1
5	0.00285	1	2 4 3 1 1
5	0.00285	1	1 2 1 2 5
3	0.13390	47	0 5 6
3	0.12251	43	0 6 5
3	0.00855	3	2 4 5
3	0.00855	3	1 4 6
3	0.00570	2	0 7 4
3	0.00285	1	1 6 4

Catalog element 4			
length	prob	accum#	string
3	1.00000	151	0 3 8

Catalog element 5			
length	prob	accum#	string
5	0.08538	94	0 4 5 1 1
3	0.91462	1007	0 4 7

the criteria followed a convergence failure by the other criterion will not negate the consequence of the first criterion. Added samples can only strengthen the argument for the consequence of either of the convergence criteria. That is, the consequences of using both criteria is at least as great as both of the consequences.

The consequences of the criterion that no new samples be added during the i-th iteration are now analyzed. In order to formalize this criterion, let Ω be the set of all strings which *can* be generated by the given pattern representation scheme. This is a finite set with cardinality j. Since any element of Ω can can be generated by any target, for a given target and pattern representation scheme there exists a probability distribution which assigns all elements of Ω to some probability, r. Hence, for each catalog element, C_i, there exists a mapping $P_{C_i}(X)$:

$$P_{C_i}(X) : X \rightarrow r \in [0,1] \quad \forall X \in \Omega$$

During execution of the density creation algorithm, a number of strings are generated each iteration. To be more exact, during the i-th iteration n strings are generated.

Let O_i be the set of unique strings which are elements of the *estimated* density at the onset of the i-th iteration. In other words, it is the set of unique strings which have been generated during the previous $(i-1)$ iterations of the algorithm. Then $\bar{O}_i$ is the set of strings which have not been generated by the algorithm at the i-th iteration. Hence:

$$O_i \cup \bar{O}_i = \Omega, \quad O_i \cap \bar{O}_i = \emptyset \tag{2.3}$$

Since the noise generation and pattern representation sections operate independently and since independent random variables are used for the noise generation process we can assume that the string generation process produces independent, identically distributed strings which have the distribution described above. Thus we can say that at the i-th iteration:

$$P(X \in \bar{O}_i) = \sum_{X \in \bar{O}_i} P(X) \ = p_i \tag{2.4}$$

$$P(X \in O_i) = \sum_{X \in O_i} P(X) \ = q_i \tag{2.5}$$

$$p_i + q_i = 1 \tag{2.6}$$

At the i-th iteration, let the k-th string generated by the noise generation and pattern representation sections be $X(k)$. Furthermore, define a random variable $Y(k)$ which is as follows:

$$Y(k) = \begin{cases} 1 & \text{if } X(k) \in \bar{O}_i; \\ 0 & \text{if } X(k) \in O_i. \end{cases}$$

Then clearly $Y(k)$ is a Bernoullian random variable and has the following distribution

$$Y(k) = \begin{cases} 1 & \text{with probability } p_i; \\ 0 & \text{with probability } q_i. \end{cases}$$

and

$$\mathbf{E}\{Y(k)\} = p_i, \quad \sigma^2(Y(k)) = p_i - p_i^2 \quad \forall k = 1, 2, \ldots, n \tag{2.7}$$

Define the random variable S_n

$$S_n = \sum_{k=1}^{n} Y(k) \tag{2.8}$$

Thus $Y(k)$ is an indicator variable for the set $\bar{O}_i$ and S_n represents the number of new strings which are added during the i-th iteration. Convergence failure by added samples can now be expressed as the condition that $S_n = 0$ after completion of some iteration of the process.

In order to characterize the implications of such a criterion let us consider the quantity $\frac{S_n}{n}$ to be an estimate of the value of p_i at the i-th iteration. Then each iteration of the process can be considered an experiment to estimate the quantity p_i. This estimate can be characterized by equation 7.6.12 of [10].

$$P\left(\left|\frac{S_n}{n} - p_i\right| \geq c\right) \leq \frac{p_i - p_i^2}{c^2 n} \leq \frac{1}{4c^2 n} \tag{2.9}$$

Which is an application of Chebyschev's inequality to a sum of n Bernoullian random variables. This equation states that the *probability* that our estimate of the parameter p_i is in error by no more than c is less than or equal to $\frac{1}{4c^2 n}$. In the case that the estimate $\frac{S_n}{n}$ is non-zero then the new strings are appended to the set O_t to make the set O_{t+1} and a the process continues with a new experiment (iteration). However, if the estimate is zero then the process halts and equation (2.9) becomes:

$$P(p_t \geq c) \leq \frac{1}{4c^2 n} \tag{2.10}$$

Since the estimate is zero this equation states that the probability that the true parameter p_t is greater than c is no more than $\frac{1}{4c^2 n}$. As a numerical example, for $n = 50$ (which was the case for the experiments in this study) equation (2.10) states that there is a 50% chance that 90% of the space is covered when the algorithm halts. Furthermore if the probability on the left side of equation (2.9) is estimated with a normal distribution (see [10] equation 7.6.14) then equation (2.9) becomes

$$P(p_t \geq c) \approx 2\left[1 - \Phi\left(\sqrt{4n}c\right)\right] \tag{2.11}$$

where

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{\frac{-u^2}{2}} du \tag{2.12}$$

Which states that there is an 85% chance that 90% of the space is covered when the algorithm halts for $n = 50$.

The problems associated with likelihood function creation are all but overshadowed by the advantages of likelihood function tests. Since many different decision rules can be expressed in terms of likelihood functions, a system which has a likelihood function available is very flexible. Thus, as a-priori or cost information becomes available, or as the nature of the desired optimality changes, the classifier can be easily modified to account for these changes. While likelihood functions are only valid for the average noise power level at which their constituent densities were constructed, (training noise) the ordering induced on strings by the densities should be invariant to a change in noise level. This implies that a decision rule which uses a likelihood function that is created at a fixed noise level exhibits robustness with respect to a change in noise level.

2.2.3 TESTING

Testing of the simulated system is done with two methods, complete and Monte-Carlo. Complete testing is not, actually, true testing. Complete testing evaluates the system performance by assuming that the system is operated exactly at the noise level with which the likelihood functions was created and that the likelihood function used is not an estimate of the true likelihood function but instead is the true likelihood function. From these assumptions a confusion matrix and percent misclassification is calculated. The percent of unknown-classification is taken to be zero since the likelihood function is exact; there is no probability that a string outside the range of the likelihood function occurs. Monte-Carlo testing, on the other hand does not depend on such assumptions. The only assumption implicit in Monte-Carlo testing is that the simulated radar data is representative of actual radar data.

If the assumption is made that $\hat{P}(x \mid G_j) = P(x \mid G_j)$ (that the likelihood function is *complete*) then the calculation of the confusion matrix used in complete testing is as follows:

$$\mathbf{P}(G_i \mid G_j) = \sum_{x \in X_j} \hat{P}(x \mid G_j) D(x, G_i) \quad \forall\, i, j \in \{1, 2, \ldots, n\} \tag{2.13}$$

Where $P(x \mid G_j)$ is the probability that string x occurs given that target j is present (the conditional density), X_i is the domain of the conditional density, restricted to target i. $\hat{P}(x \mid G_j)$ is the estimate of the probability that string x occurs given that target j is present. $D(x, G_i)$ is the decision rule indicator function, $D(x, G_i) = 1$ if the given decision rule indicates target i on string x, else $D(x, G_i) = 0$. $\mathbf{P}(G_i \mid G_j)$ is the probability that target i is declared given that target j is actually present. (this is the i-th element of the confusion matrix on the j-th row). n is the number of catalog elements.

Complete testing does provide a somewhat adequate estimate of the true confusion matrix but Monte-Carlo testing does a better job since the assumptions implicit in Monte-Carlo testing are more justifiable. Complete testing, on the other hand, does give an indication as to how well the classes separate (classes, in this case, being defined as the ranges of the densities which make up the likelihood function.) As such, complete testing provides some information about system performance.

For Monte-Carlo testing, one hundred trials are run for each noise level tested. This number is taken from previous studies which indicate one hundred is statistically sufficient [11]. To simulate the noise, independent Gaussian random variables are added to the in-phase and quadrature components of the noiseless radar measurement data. The random variables have zero mean and their variance depends upon the desired noise level. In addition to confusion matrices and percent misclassification, percent unknown-classification are computed.

2.3 RESULTS

2.3.1 MONTE-CARLO SIMULATION RESULTS

Execution times required for likelihood function creation can range from several minutes to days. The purpose of this portion of the study is to demonstrate the feasibility of implementing a syntactic pattern recognition system for radar target identification and to investigate the properties of the various pattern representation schemes rather than to provide a comprehensive performance evaluation of the classifier. For these reasons, experimental results are provided for only a few, nominal, scenarios. Performance for other cases *may* be extrapolated from the performance of other decision-theoretic classifiers. However, the pattern representation schemes are highly non-linear processes and the classification performance of these systems may not exactly parallel that of the more traditional classifiers.

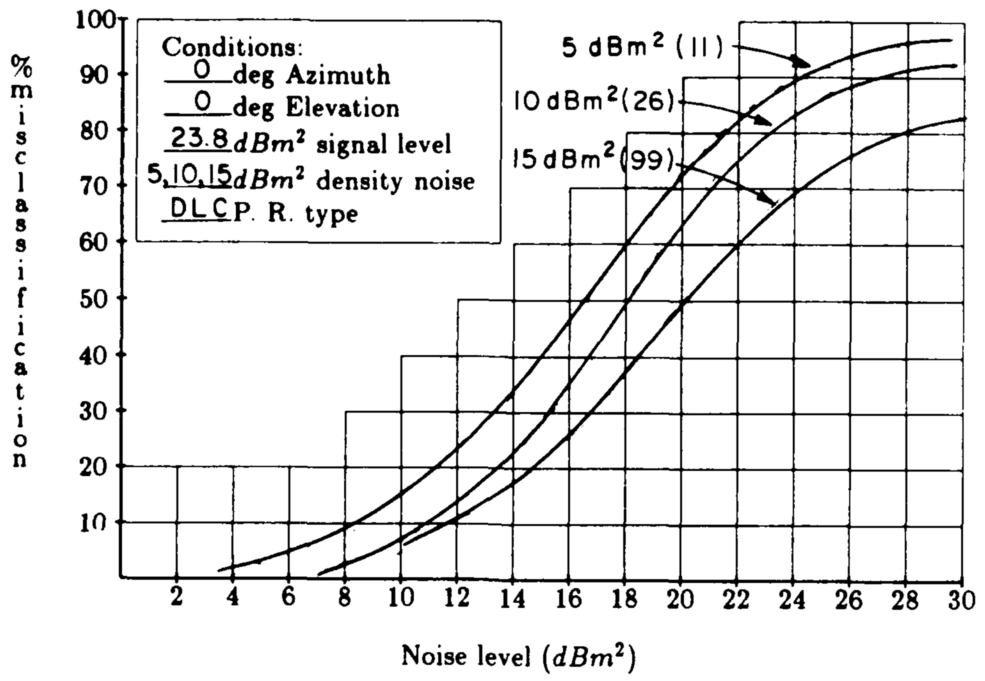


Figure 5: Double level crossing test results

Figures 5, 6, and 7 represent results of tests run on a catalog of five radar sampled frequency spectra. The cataloged sampled frequency spectra represent the radar returns of five different aircraft at an aspect of 0° azimuth, 0° elevation. 11 frequencies are used from 8 to 58 MHz, spaced by 5 MHz using HH polarization. The average signal level is 23.8 dBm^2. 100 trials per target are used for the Monte-Carlo simulation in all cases. The different curves on the graphs are labeled with the noise used for likelihood function creation (training noise) and a number which indicates the amount of memory needed to store the likelihood function, in blocks. Note that increase performance is achieved with increased *training* noise and not with increased test or *system* noise.

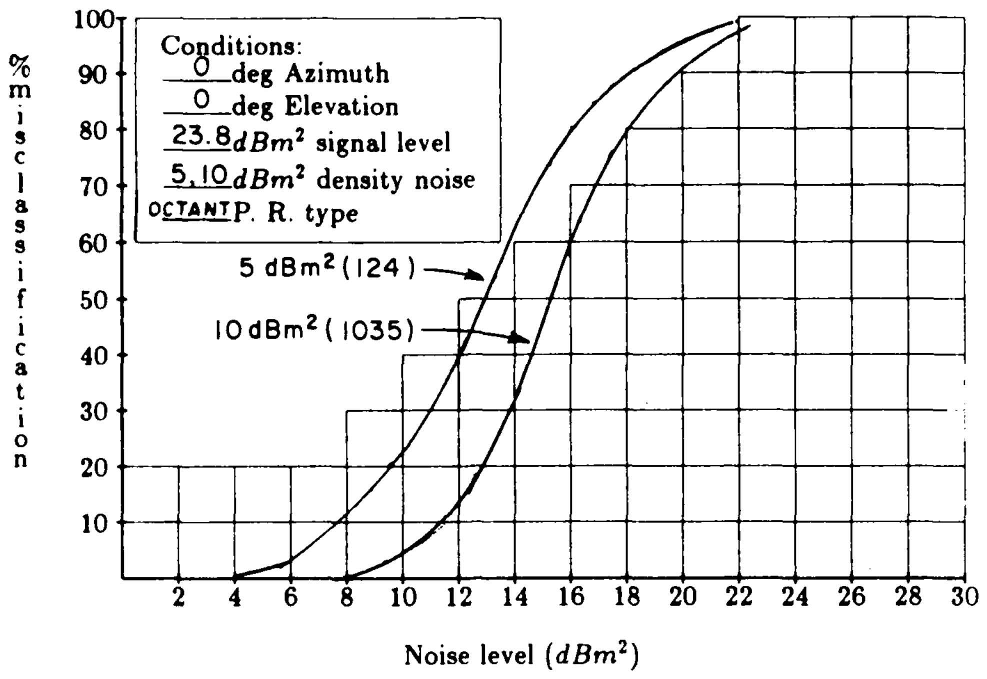


Figure 6: Octant crossing test results

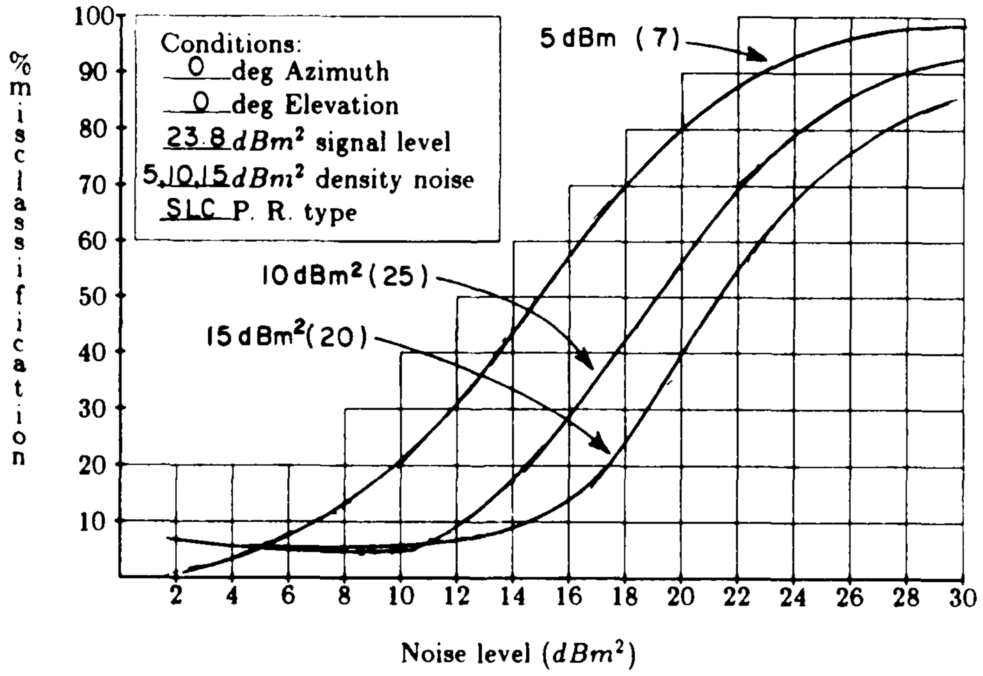


Figure 7: Single level crossing test results

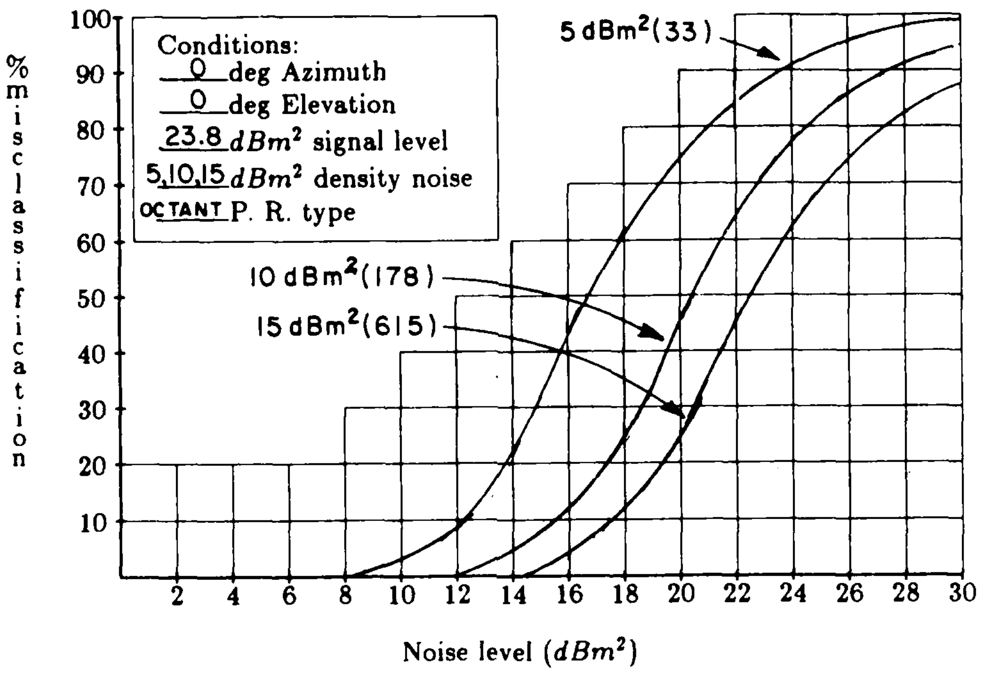


Figure 8: Octant crossing test results (6 frequency)

Figure 8 represents results of tests run on the same catalog of five aircraft radar returns described above, the only difference being that six instead of eleven frequencies were used. Again, the frequencies were equally spaced from 8 to 58 MHz. Curve labeling does not change.

Figures 9, 10, and 11, represent results of tests run on the same catalog of five aircraft radar returns. 11 frequencies are used for the single level crossing and double level crossing pattern representation schemes while six frequencies are used for octant crossing. This is done since the results of the eleven frequency case for octant crossing are disappointing and the likelihood function storage requirements were out of bounds. The training noise remains constant on each of the graphs

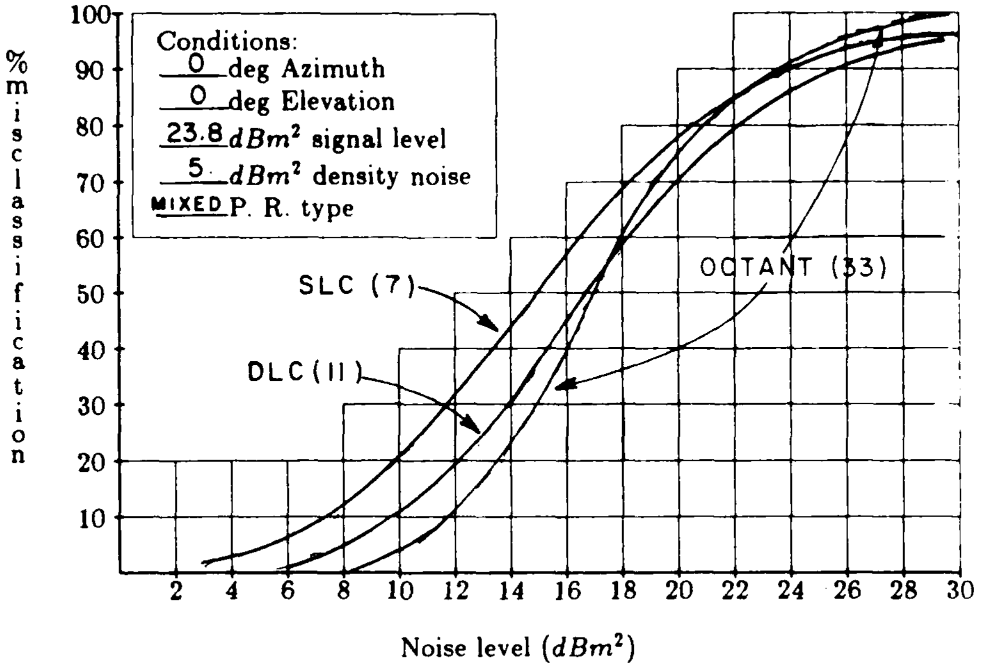


Figure 9: Fixed Likelihood function noise level ($5dBm^2$)

while the different curves on the graphs are labeled with their corresponding pattern representation scheme. Again, the number of blocks required to store the likelihood function is given in parentheses.

2.3.2 OBSERVATIONS

As with all classification systems, increasing test noise increases average percent misclassification in a monotonic sense. Also an increase in the test noise increases percentage unknown classification (see Chapter III.) Comparison with other classifiers should include a correction to the percent misclassification by decreasing it by $\frac{1}{n}$ of the percent unknown-classification to account for a randomized decision rule (as described previously) in the case when the system is unable to

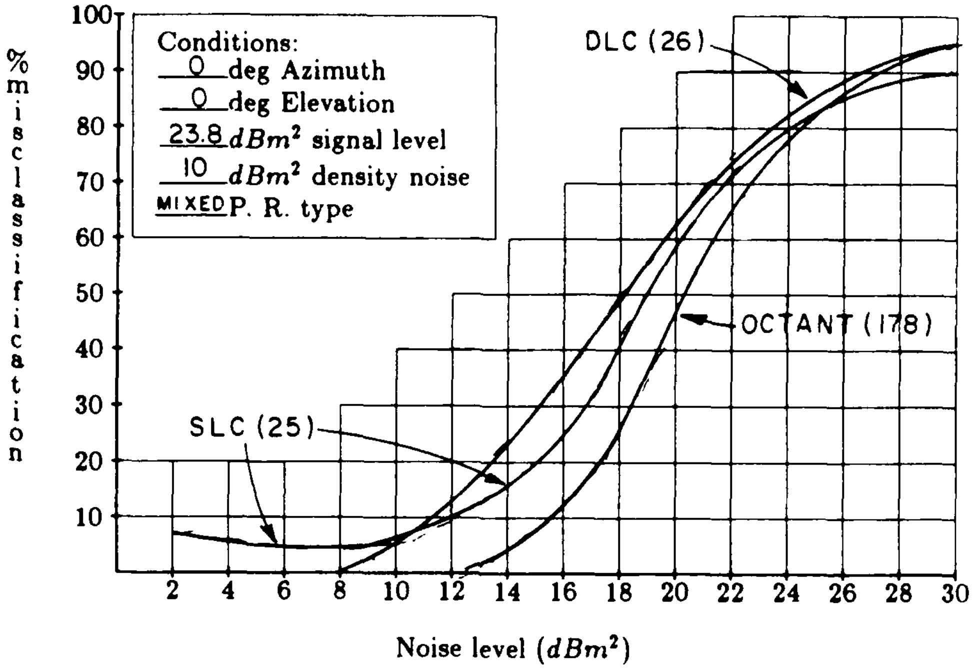


Figure 10: Fixed Likelihood function noise level ($10dBm^2$)

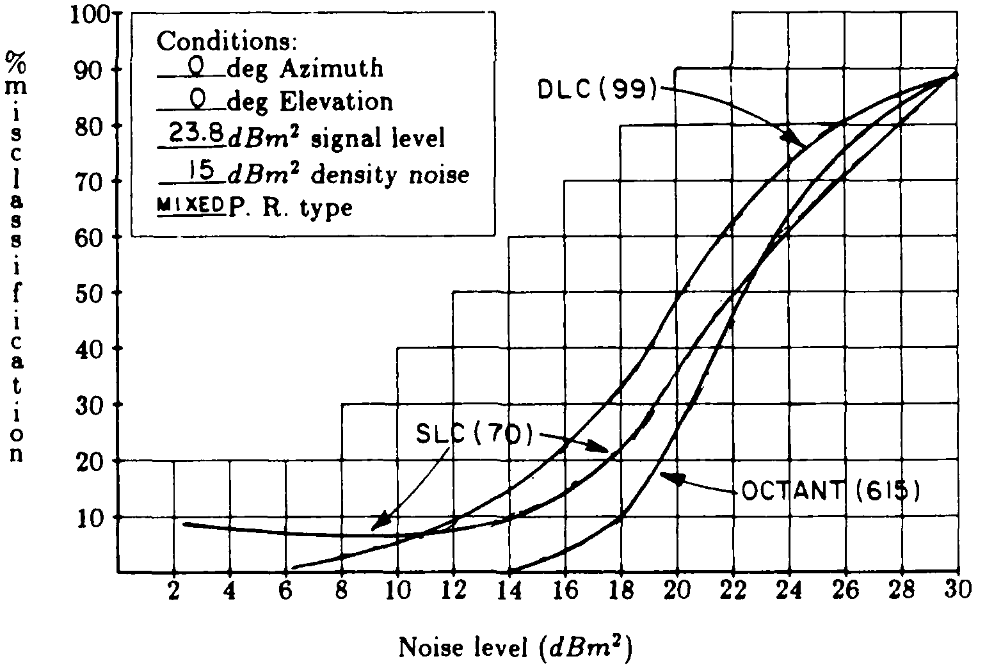


Figure 11: Fixed Likelihood function noise level ($15dBm^2$)

classify.

Increasing training noise decreases percent misclassification at test noise levels which are above the training noise level considerably for double level crossing and octant crossing pattern representation schemes. However, single level crossing performance suffers at low levels with an increase in training noise level. An increase in the training noise level has more of an effect than changing pattern representation scheme. However, likelihood function storage requirements increase non-linearly with an increase in training noise level.

Single level crossing exhibits a matched noise phenomenon to a much higher degree than double level crossing. That is, for a given noise level, the decision rule formed by the likelihood function made for that noise level (training noise) is significantly better than any decision rule that uses a likelihood function made at a different noise level. This can be seen from the fact that the single level crossing curves are below the other single level crossing curves at their given training noise levels. The double level crossing schemes exhibit this phenomenon also but to a lesser degree. Octant crossing appears to exhibit no matched noise phenomenon.

The performance of the octant crossing scheme using six frequencies is significantly better at low test noise levels, than other schemes. This is true at all training noise levels. However, octant crossing likelihood functions storage requirements are much larger than single level crossing and double crossing. In addition, single level crossing performance at high noise levels is comperable to octant performance using 15 dBm^2 training noise.

CHAPTER III

SYNTAX ANALYSIS SIMPLIFICATION

3.1 INTRODUCTION

Chapter two established the feasibility of using a practical pattern representation scheme to form a syntactic pattern recognition system for radar target identification. In this chapter a likelihood function from the previous chapter is used in conjunction with a fixed decision rule to derive a set of non-deterministic non-stochastic finite state automata. These automata implement and, under certain circumstances, extend, the given decision rule, restricted to the likelihood function estimate.

3.2 LANGUAGE THEORY

Some definitions from language theory are now given for notational and uniformity purposes. The definitions are taken from various pattern recognition and language theory texts (see [5], [12], [13].) Since string languages are the only languages dealt with in this study, definitions pertinent to string grammars are given.

String A string is a concatenation of elements of a given set. The set from which the elements are taken is called the alphabet. For example, the set $\{0,1\}$ is the alphabet for the string 0100101. Juxtaposition of two strings represents concatenation.

Length The length of a string is the number of elements from the alphabet in the string. The empty string is the string with length 0, it is denoted by the letter e.

Closure A set T^* is the set of all strings generated by using the set T as an alphabet. The set T^+ is the set T^* less the empty string e.

Language A language is a set of strings over some alphabet.

Regular Language A regular language is a language which has one of the following forms:$\{\}, \{e\}, \{a\}$ (where a is any member of the alphabet), $R_1 \cup R_2$, $R_1 \cap R_2$, R_1^* where R_1 and R_2 are regular languages.

Finite state automaton A finite state automaton is a recognizer of regular languages. It can be thought of as a state diagram, such as is used in digital design of sequential machines, with the added feature of non-determinism and an extended symbol set. More formally, a finite state automaton is a 5-tuple, $A = \{Q, T, P, S, F\}$ where

Q is the set of states of the automaton.

T is the set of symbols from which input strings are formed.

P is the set of transition mappings which define how the automaton moves from state to state. It is a mapping from $Q \times T \mapsto Q$.

S is the initial state of the automaton.

F is a subset of Q which defines the final states of the automaton. An input string is said to be accepted by an automaton if the operation of the automaton terminates in one of the final states.

Stochastic finite state automaton A *stochastic* finite state automaton is simply a finite state automaton with the added feature that each of the elements of the transistion function is assigned a probability. More formally, a stochastic finite state automaton is a 6-tuple, $A = \{Q, T, P, S, F, R\}$ where Q, T, P, S, F are defined as in the non-stochastic case and R is the set of probabilities associated with the elements of the transistion function P.

3.3 GRAMMATICAL INFERENCE

3.3.1 PURPOSE

Grammatical inference techniques derive a grammatical system from a given sample of sentences (strings.) The given sample consists of two sets of strings (see [14].) One set, which the grammatical system must accept, is called the positive sample or R^+. The other set, which mu t be rejected by the inferred system is called the negative sample or R^-. Some inference techniques include the negative sample in their calculations, others do not.

3.3.2 K-TAILS METHOD

The k-tails method of grammatical inference derives a non-deterministic non-stochastic finite state automaton from a positive sample and a given value of k. The negative sample is not used in the k-tails inference technique. The value of k determines the nature of the language accepted by the automaton as described below. The accepted language can be controlled in such a way as to reject the negative sample.

The following is a summary of Bierman and Feldman's k-tails method from [15] as described in [5]. In this method, the set of terminals (primitives) is assumed to be known, let V_t be the set of terminals. The k-*tail* of a string z with respect to a set of strings, R^+ is:

$$h(z, R^+, k) = \{x \in V_t^* \mid zx \in R^+ \textit{ and } |x| \leq k\}$$

Where the term zx denotes the concatenation of the two strings z and x. Two k-tails, $h1, h2$, are equal if $h1 \subset h2$ and $h2 \subset h1$. Two strings are said to be k-tail equivalent if their k-tails are equal.

The automaton generated from the positive sample R^+, generated by the k-tails method is: $A = \{Q, T, P, S, F\}$ where,

$Q = \{q \in 2^{V_t^*} \mid h(z, R^+, k) = q$ for some $z \in V_t^*\}$,

$T = V_t$,

$P(q,a) = \{\acute{q} \in Q \mid \text{for all } z \in V_t^* \text{ such that } h(z,R^+,k) = q \text{ and } h(za,R^+,k) = \acute{q}\}$,

$S = \{q_1 \in Q \mid h(e,R^+,k) = q_1\}$,

$F = \{q \in Q \mid e \in h(q,R^+,k)\}$.

This definition of the k-tails method is adequate for making a rigorous definition and proofs about the nature of an automaton resulting from the technique. However, the method requires adaptation for implementation on a digital computer. Since the closure of the set of terminals is infinite, the adaptation begins by constructing a finite subset of V_t^* which are the only elements of V_t^* which *may* produce non-empty k-tails.

$$D_s = \{z \in V_t^* \mid zx \in R^+, \text{for some } x \in V_t^*\} \tag{3.1}$$

The actual automaton produced is now : $A = \{Q,T,P,S,F\}$ where,

$Q = \{q \in 2^{V_t^*} \mid h(z,R^+,k) = q \text{ for some } z \in D_s\}$,

$T = V_t$,

$P(q,a) = \{\acute{q} \in Q \mid \text{for all } z \in D_s \text{ such that } h(z,R^+,k) = q \text{ and } h(za,R^+,k) = \acute{q}\}$,

$S = \{q_1 \in Q \mid h(e,R^+,k) = q_1\}$,

$F = \{q \in Q \mid e \in h(q,R^+,k)\}$.

Increasing the value of k reduces the number of D_s entries which are k-tail equivalent and, thus increases the number of resulting states. More states produce a more complex automaton and thus we can say that automaton complexity is proportional to the value of k chosen. Fewer states limit the *control* which can be placed on the accepted language ($L(G)$). This implies that the inference process will produce a relatively complex automaton which accepts a restricted language when a large value of k is used and a relatively simple automaton which accepts a relatively unrestricted language when a small valude of k is used. Bierman and Feldman have formalized this notion with a relation between languages accepted by automata generated with different values of k. Let L_1 be the language accepted by an automaton which was generated with $k = k_1$ and L_2 be the language accepted by an automaton which was generated with $k = k_2$, let k_{max} be the length of the longest string in the positive sample, R^+.

$$L_2 \subseteq L_1 \quad \text{if} \quad k_1 < k_2 \tag{3.2}$$

$$L_1 = R^+ \quad \text{if} \quad k_1 = k_{max} \tag{3.3}$$

The domain of the likelihood function example given in Chapter 2, restricted to catalog element 2, is used as the positive sample in a k-tails derivation of a grammar using $k = 1$. The resulting D_s set and state assignment are given in Table 5.

Table 5: D_3 state assignment

D_3 element	State assignment
e	$\{\}$
1	$\{\}$
1 1	$\{\}$
1 1 1	$\{\}$
1 1 1 1	$\{\}$
1 1 1 1 3	$\{\}$
1 1 1 1 3 1	$\{3\}$
1 1 1 1 3 1 1	$\{2\}$
1 1 1 1 3 1 1 1	$\{1\}$
1 1 1 1 3 1 1 1 1	$\{e\}$
1 1 1 2	$\{\}$
1 1 1 2 2	$\{\}$
1 1 1 2 2 1	$\{\}$
1 1 1 2 2 1 1	$\{2\}$
1 1 1 2 2 1 1 1	$\{1\}$
1 1 1 2 2 1 1 1 1	$\{e\}$
1 1 1 1 3 1 1 2	$\{e\}$
1 1 1 1 3 1 2	$\{1\}$
1 1 1 1 3 1 2 1	$\{e\}$
1 1 1 2 2 1 1 2	$\{e\}$
1 1 1 2 2 1 2	$\{1\}$
1 1 1 2 2 1 2 1	$\{e\}$
1 1 1 1 3 1 3	$\{e\}$
1 3	$\{\}$
1 3 3	$\{\}$
1 3 3 1	$\{3\}$
1 3 3 1 1	$\{2\}$
1 3 3 1 1 1	$\{1\}$
1 3 3 1 1 1 1	$\{e\}$
1 4	$\{\}$
1 4 2	$\{\}$
1 4 2 1	$\{\}$
1 4 2 1 1	$\{2\}$
1 4 2 1 1 1	$\{1\}$
1 4 2 1 1 1 1	$\{e\}$
1 1 1 1 5	$\{2\}$
1 1 1 1 5 1	$\{1\}$
1 1 1 1 5 1 1	$\{e\}$
1 3 3 1 1 2	$\{e\}$
1 3 3 1 2	$\{1\}$
1 3 3 1 2 1	$\{e\}$
1 1 1 1 5 2	$\{e\}$
1 1 1 1 6	$\{1\}$
1 1 1 1 6 1	$\{e\}$
1 4 2 1 1 2	$\{e\}$
1 1 1 2 5	$\{1\}$
1 1 1 2 5 1	$\{e\}$
1 4 2 1 2	$\{1\}$

Table 5: D_s state assignment (cont)

D_s element	State assignment
1 4 2 1 2 1	$\{e\}$
1 3 3 1 3	$\{e\}$
1 3 5	$\{2\}$
1 3 5 2	$\{e\}$
1 3 6	$\{1\}$
1 3 6 1	$\{e\}$
1 4 5	$\{1\}$
1 4 5 1	$\{e\}$

with the start state being the state $\{\}$ ($S = \{\}$)
and the set of final states being the state $\{e\}$ ($F = \{e\}$)

Table 6: Recognizer example transition function

From state	on symbol	to state
$\{\}$	1	$\{\}$
$\{\}$	1	$\{3\}$
$\{\}$	1	$\{2\}$
$\{\}$	2	$\{\}$
$\{\}$	2	$\{1\}$
$\{\}$	3	$\{\}$
$\{\}$	4	$\{\}$
$\{\}$	5	$\{2\}$
$\{\}$	5	$\{1\}$
$\{\}$	6	$\{1\}$
$\{3\}$	1	$\{2\}$
$\{3\}$	2	$\{1\}$
$\{3\}$	3	$\{e\}$
$\{2\}$	1	$\{1\}$
$\{2\}$	2	$\{e\}$
$\{1\}$	1	$\{e\}$

3.4 RECOGNIZER GENERATION

An algorithm is now developed which exploits the properties of the k-tails method to derive a set of finite state automata which can be used for syntax analysis.

3.4.1 DESCRIPTION OF ALGORITHM

A fixed decision rule using a fixed likelihood function determines a partition of the observation space. These partitions can be implemented with a set of finite state automata. Thus a syntax analyzer which implements the semi-optimal decision rules of Chapter 2 can be derived by using the k-tails method to infer an automaton for each partitioned area (as defined by the decision rule) of the observation space. Each of the automata could be constructed to accept only those strings which are contained in the partition area to which they correspond. However, there is really no reason to restrict the accepted languages of the automata so severely. As long as each of the automata accepts the strings in its partition area and does not accept the strings in any of the other partition areas it would only be accepting strings outside all of the partition areas. Those strings outside the union of the partition areas are, in reality, of no concern since they are of unknown classification with respect to the likelihood function estimate.

The first step in automaton derivation is to determine the sample. The positive sample for each of the catalog elements is defined by the decision rule and a number called the *probability overlap factor* denoted as P_{of}. The probability overlap factor defines the amount of *overlap* which we are willing to tolerate among the various elements of the catalog. The positive sample for the ith catalog element is constructed by applying a test to each string in the ith density. An element of the density, x, is included in the positive sample if:

$$P_c(x, G_i) \geq (\max\{P_c(x, G_j), j = 1, 2, 3, \ldots n\})(1 - P_{of}), \tag{3.4}$$

Where, $P_c(x, G_i)$ is the comparison quantity of string x given catalog element G_i. For a maximum likelihood decision rule $P_c(x, G_i)$ is the conditional probability, for maximum a-posteriori $P_c(x, G_i)$ is the a-posteriori probability, for a Bayes decision rule $P_c(x, G_i)$ is the negative of the a-posteriori risk.

The test defined by equation (3.4) is applied to every string in the given density. The purpose of the probability overlap factor is to account for inaccuracies in the likelihood function estimate and expand the positive sample.

The negative sample, for each automaton, is the set of strings which the automaton should not accept. For obvious reasons the negative sample is set to be the union of the domains of the other densities which are not in the positive sample. This is a clear choice since the automaton must accept all the elements of the positive sample and must not accept any of the elements from the other densities since they are assigned to another catalog element. The strings which are outside the domain of the union of the densities are considered *don't care* strings. They are used to minimize automaton complexity.

Once the positive and negative sample have been determined a trial automaton is generated with $k = 1$. Since the k-tails inference process does not consider the negative sample, the generated automaton is checked to see if it rejects the entire negative sample. If the negative sample is completely rejected the next element in the catalog is processed. If the negative sample is not completely rejected then k is incremented, a new automaton is generated and checked for rejection of the negative sample. This process continues until an automaton is found which accepts the positive sample and rejects the negative sample. If k is incremented to the point that it is the length of the longest string in the positive sample then the generated automaton, at that point, accepts only the positive sample. Since the positive sample and the negative sample are guaranteed to be disjoint, a solution is guaranteed for this circumstance.

3.4.2 ALGORITHM PROPERTIES

This algorithm can be summarized as a linear search for the smallest value of k which produces an automaton which accepts the positive sample and rejects the negative sample. Thus the generated automata are minimized in complexity since they are generated under a minimum value of k. By the properties of the k-tails method given in inequality (3.2), it should be apparent that since the value of k is minimized, the accepted language is maximized. That is, the set of strings accepted by an automaton generated with the minimum value of k contains the strings which are accepted by automata generated with larger values of k. This maximization of the accepted language has the effect of extending the decision rule beyond that defined by the likelihood function test. It is hoped that this process is able to extend the decision rule in a way which has increased robustness over the likelihood function decision rule. It must be kept in mind that the minimization of the automata is with respect to the other catalog elements and, as such, the process must be repeated with every new set of catalog elements.

Different values of the probability overlap factor produce different sets of automata. With $P_{of} = 0$ the generated automata *exactly* implements the decision rule specified by the likelihood function test, (when restricted to the domain of the likelihood function.) A non-zero value of P_{of} implies that a string may be in the positive sample of more than one of the automata. This implies that the decision rule specified by the automata set could possibly declare more than one catalog for such a string. The testing section of the simulation accounts for such instances by simulating a randomized decision rule for such a case. This is done for normalization purposes so that the results of the likelihood function tests can be compared with their automaton implementations. As stated before, increasing the value of P_{of} expands the positive sample. The effect of expanding the positive sample in this manner is a decrease in automaton complexity. This is expected since fewer strings from the domain of the density are excluded from the positive sample. If it is assumed that strings from the domain of a given density have a similar structure then the added data should allow the k-tails algorithm to infer a less complex automaton since fewer "exceptions" must be made.

3.5 RECOGNIZER RESULTS

3.5.1 GRAPHS

Complete testing, as described in Chapter 2, was performed on the decision rules formed by the automata sets. With the probability overlap factor set to zero, confusion matrices generated by the tests run on the systems which used automata sets were identical to the confusion matrices run on systems which use the corresponding likelihood function tests. Thus the assertion that the automata sets implement the same decision rules as the likelihood function tests when restricted to the domain of the likelihood function is confirmed. As an example, the confusion matrices for a maximum likelihood function test using the likelihood function given in Chapter 2 and a corresponding automaton set are now given.

The ML confusion matrix

0.823920	0.000000	0.176080	0.000000	0.000000
0.000000	1.000000	0.000000	0.000000	0.000000
0.028490	0.000000	0.971510	0.000000	0.000000
0.000000	0.000000	0.000000	1.000000	0.000000
0.000000	0.000000	0.000000	0.000000	1.000000

The automaton set confusion matrix

0.823920	0.000000	0.176080	0.000000	0.000000
0.000000	1.000000	0.000000	0.000000	0.000000
0.028490	0.000000	0.971510	0.000000	0.000000
0.000000	0.000000	0.000000	1.000000	0.000000
0.000000	0.000000	0.000000	0.000000	1.000000

As in chapter two, Monte-Carlo testing was performed on the simulated system. From the tests, percent misclassification and percent unknown classification as a function of noise level was estimated. This data is given in graphical form. The cases which were run are the cases given in Chapter 2. One hundred trials per target were used for the Monte-Carlo simulation in all cases. The different curves on the graphs are labeled with the value of P_{of}. The results of Monte-Carlo testing done on the simulated system which used likelihood function estimates is given for comparison purposes. The likelihood function system test results are labeled with the decision rule used. As before, a number which indicates the amount of memory needed to store the likelihood function, or automaton (as the case may be) is given in blocks.

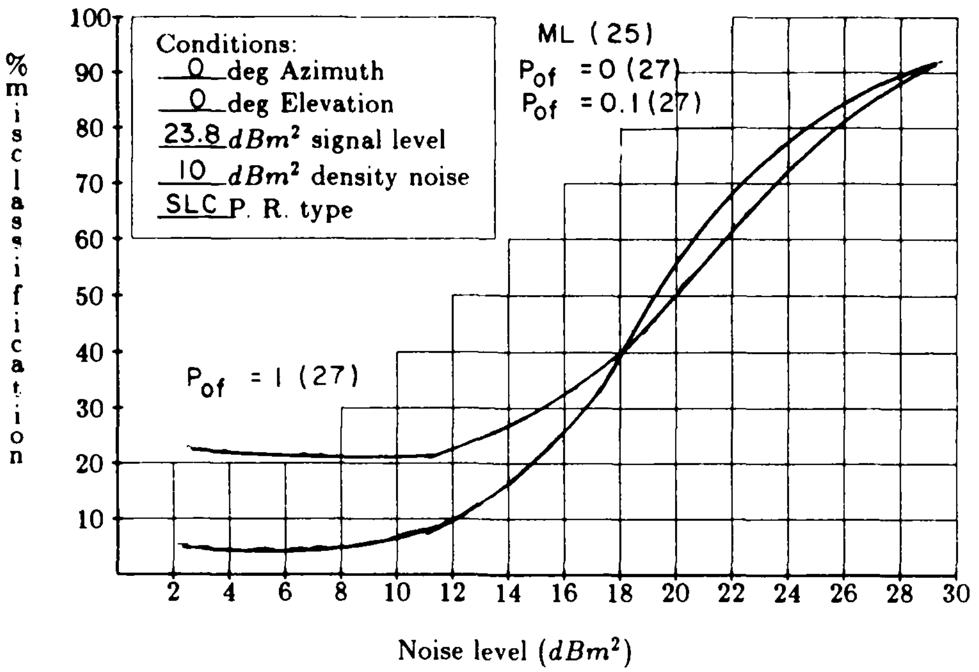


Figure 12: Single level crossing automaton comparison using 10 dBm^2 likelihood function

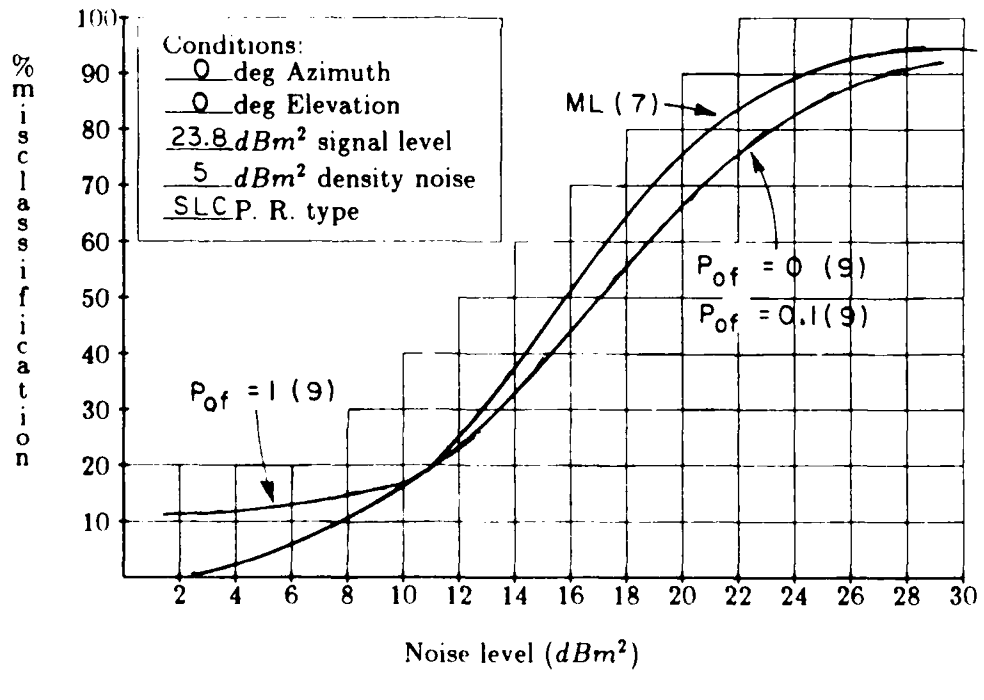


Figure 13: Single level crossing automaton comparison using 5 dBm^2 likelihood function

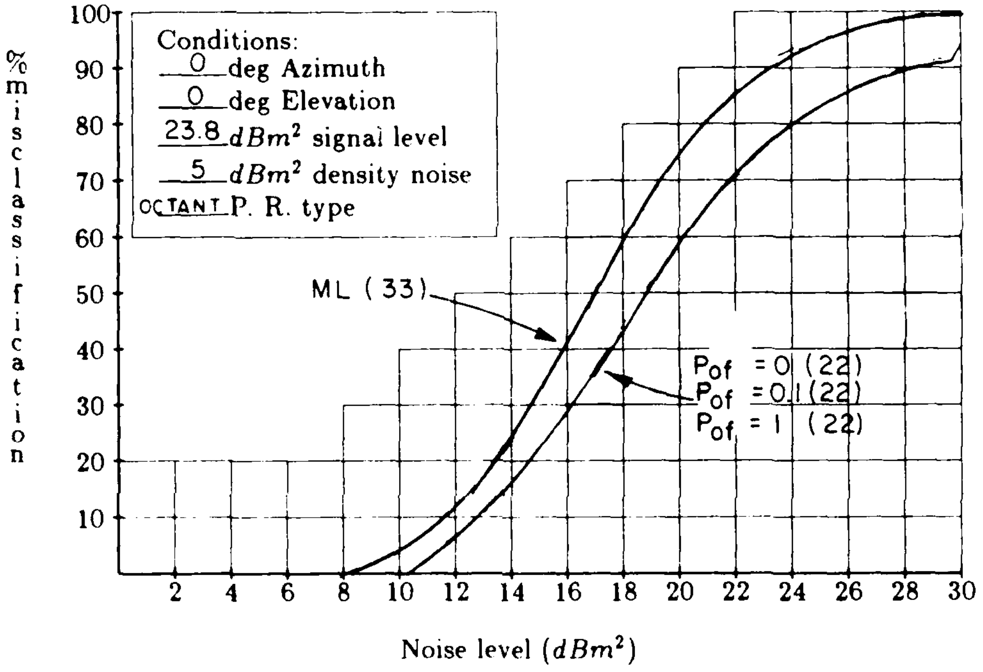


Figure 14: Octant crossing automaton comparison using 5 dBm^2 likelihood function

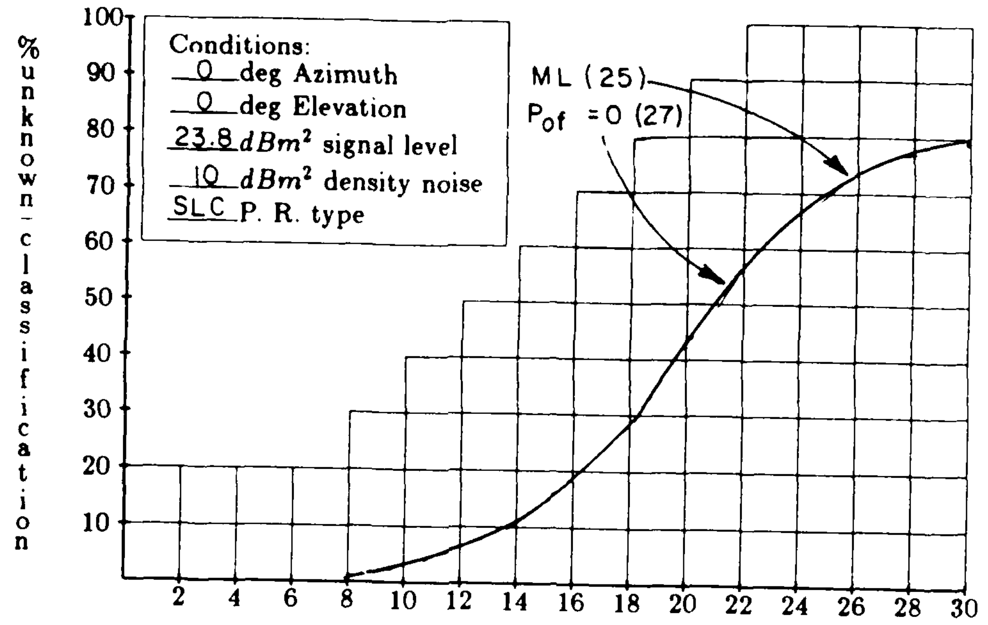


Figure 15: Percent unknown classification, SLC, 10 dBm^2 likelihood function and associated automata

3.5.2 OBSERVATIONS ABOUT RESULTS

The graphical data shows that the finite state automata decision rules are able to classify targets about as well as likelihood function tests. With the exception of the $P_{of} = 1$ curves the automaton performance curves were, very close to the likelihood function test curves. The differences in most of the curves could be attributed to the estimation error of the Monte-Carlo testing.

Under certain circumstances, the automata did not work as well as the likelihood function tests. Increasing the probability overlap factor, P_{of}, had the effect of increasing automaton size (number of states) and significantly increasing percentage mis-classification. The expected effect of decreasing automaton size by

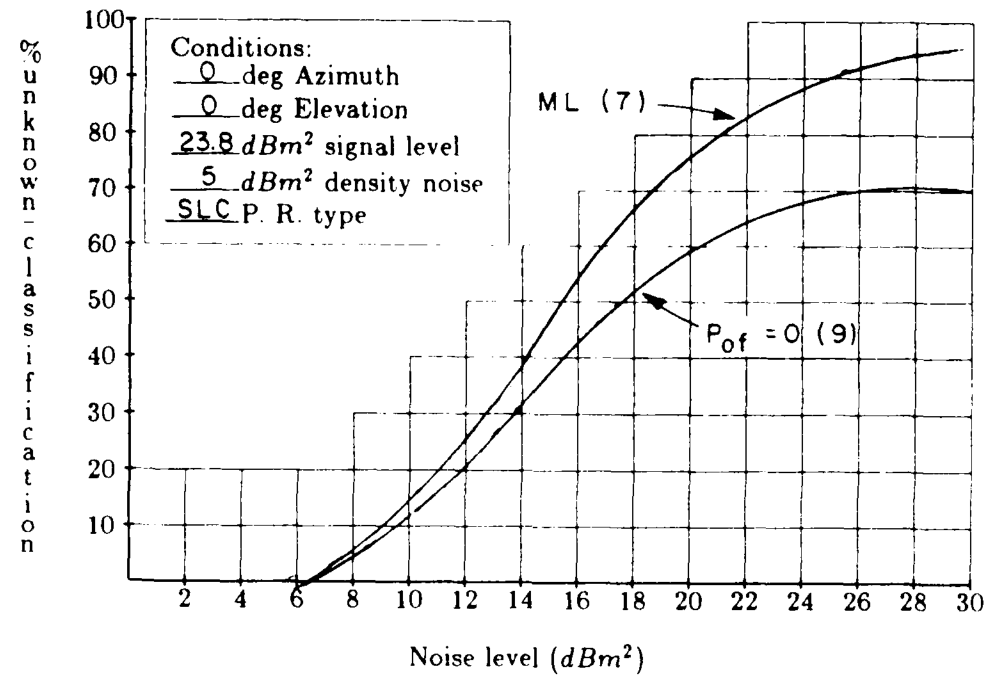


Figure 16: Percent unclassification, SLC, 5 dBm^2 likelihood function and associated automata

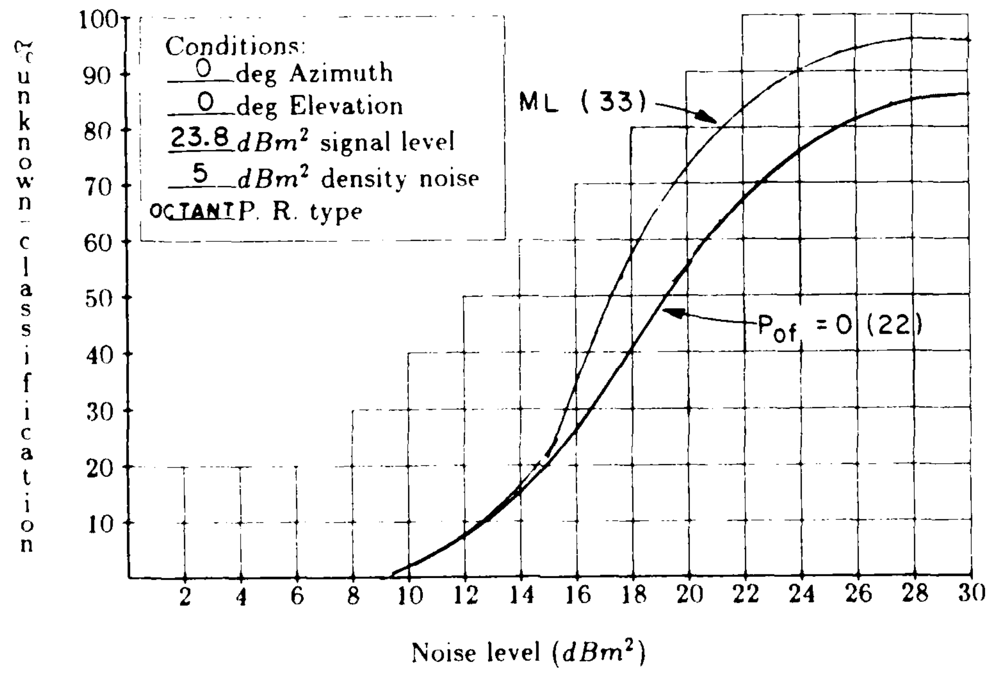


Figure 17: Percent unclassification, Octant crossing, 5 dBm^2 likelihood function and associated automata

Table 7: Automaton example statistics

Cat no.	# of states	# of trans.	k used	k max
1	12	28	2	9
2	6	19	1	9
3	22	43	3	9
4	4	4	1	3
5	5	8	1	5

increasing P_{of} was not realized. This indicates that the similarity of strings from the domain of a likelihood function is not derivable by this method of grammatical inference. The degraded classification performance comes from the fact that at $P_{of} = 1$, all of the *probability* information contained in the likelihood function estimates is being discarded and only the information contained in the domain of the individual densities is being used.

Examination of the derived automata shows that their complexity is significantly reduced by minimizing the value of k. This is evidenced by a typical automaton statistics compilation for an automaton generated for the likelihood function example of Chapter 2.

In Table 7 notice that the value of k actually used in automaton generation is significantly below the maximum value of k which would be needed in a worst case condition. However, in most cases, the percent unknown-classification of the automata syntax analysis is not significantly different from the percent unknown-classification of the likelihood function tests. This is because the strings that the inference process has used to simplify the resulting automata are outside the set of strings generated by the pattern representation scheme. In cases when the percent unknown-classification of the automata is significantly less than the percent unknown-classification of the corresponding likelihood function test the automaton does significantly better than the likelihood function test. This improvement is beyond the improvement possible with a randomized decision rule. Thus, under certain circumstances, the inference process is able to deduce the structure inherent in the densities and extend the decision rule in a way which reduces misclassification.

CHAPTER IV

SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

This study indicates that the information content of pattern representation schemes which describe the structure of the *sampled frequency spectra* of a radar return from a target is sufficient to classify aircraft targets. The structural descriptions of the sampled frequency spectra are the level crossing based pattern representations of Chapter II. This conclusion may be drawn from the set of tests performed on a classifier, which uses only a symbolic pattern representation, as discussed in Chapter II.

The pattern representation scheme that includes phase information produces the most favorable results at low noise levels. Previous studies [11] have indicated that phase information aids the classification process at low noise levels. On the other hand, the classifier which uses phase information, as well as the coherent classifiers in similar studies, become confused at high noise levels. This observation does not imply that phase information reduces the information content, rather it indicates that the classifier is suboptimal in some respect. The performance of the other pattern representation schemes as indicated in Chapter II was satisfactory, and better than the performance of the coherent scheme in certain cases.

In Chapter II it is indicated that the performance of a classifier using the single level crossing pattern representation scheme is best when that classifier uses a likelihood function which is created with the test noise level. For the single level crossing scheme, an optimal classifier must use knowledge of the noise level to choose a likelihood function for classification. However, better *overall* performance can be achieved by increasing the noise level used for likelihood function creation in the coherent scheme and the double level crossing scheme. This indicates that knowledge of the noise level is not necessary for optimal classification when using these pattern representation schemes.

The results from the feasibility study indicate that a syntactic system that *uses a practical pattern representation scheme could be constructed*. Grammatical inference techniques are used to produce a syntax analysis system that uses finite state automata to classify a target using only a symbolic structural description. The resulting classifier has a simpler implementation than the likelihood function test and represents the decision rule succinctly. Under certain circumstances, the inference process is able to extend the decision rule from the information contained in the likelihood functions. Even though the circumstances when this occur are anomalous (very low noise creation level) this effect implies that the inherent structure of the sampled frequency spectra is represented in the likelihood functions.

Three avenues of further investigation are indicated by the results of this study. The performance of the decision rules based on likelihood functions clearly is dependent upon the accuracy of the likelihood function estimate. The consequences of inaccuracy in the likelihood function estimate has not been investigated thoroughly. Implementation of a more complex syntax anal[illegible]r, such as one that uses context-free or context-sensitive grammars, could realize better performance since the inherent structure a sampled frequency spectra could be represented more accurately and succinctly with a more complex grammar. This, in turn, would produce a system that exhibits robustness at higher noise levels. However, the problems associated with more complex grammatical systems, such as a more complex inference process and classifier, might hinder the development of such a system. Development of pattern representation schemes which use strings of symbols which are directly related to the geometric structures of the targets is also indicated. A pattern representation scheme producing primitives which describe the geometric structure of the target could realize the full potential of syntactic methods; the ability to describe the target structurally. Direct extensions of this study would include a more careful examination of the likelihood function estimate, inference of a more sophisticated syntax analyzer as well as investigation of better pattern representation schemes.

Further extensions of this study could investigate the feasibility of using higher-dimensioned pattern representation schemes. Obviously, to *fully* describe the geometric structure of a target, a higher dimensional pattern representation scheme is needed. Other studies [6] have suggested that classification could be better achieved through tree representation of the sampled frequency spectra instead of using a string representation. Thus the investigation of higher dimensioned pattern representation schemes appears to be both promissing and challenging.

REFERENCES

[1] A. Kamis, F. D. Garber, and E. K. Walton, "Radar target classification studies - software development and documentation," Technical Report 716559-1, The Ohio State University, Department of Electrical Engineering, Electro-Science Laboratory, September 1985.

[2] A. A. Ksienski, Y. T. Lin, and L. J. White, "Low-frequency approach to target identification," *Proceedings of the IEEE*, vol. 63, no. 12, pp. 1651-1660, December 1975.

[3] M. I. Skolnik, *Introduction to Radar Systems.* New York: McGraw-Hill, 1980.

[4] E. K. Walton and J. D. Young, "The Ohio State University compact radar cross-section measurement range," *IEEE Transactions on Antennas and Propagation*, vol. AP-32, no. 11, pp. 1218-1223, November 1984.

[5] K. S. Fu, *Syntactic Methods in Pattern Recognition.* New York: Academic Press, 1974.

[6] Y. Cheng and S. Lu, "Waveform correlation by tree matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, no. 3, pp. 299-305, May 1985.

[7] B. F. Logan, Jr., "Information in the zero crossings of bandpass signals," *Bell System Technical Journal*, vol. 56, no. 4, pp. 487-510, April 1977.

[8] S. R. Curtis, A. V. Oppenheim, and J. S. Lim, "Signal reconstruction from Fourier transform sign information," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-33, no. 3, pp. 643-656, June 1985.

[9] F. D. Garber and O. Snorrason, "Nonparametric discriminant analysis applied to feature selection for radar target identification," (in preparation).

[10] K. L. Chung, *Elementary Probability Theory with Stochastic Processes.* New York: Springer-Verlag, 1974.

[11] N. F. Chamberlain, "Surface ship classification using multipolarization, multifrequency sky-wave resonance radar," M.S. Thesis, The Ohio State University, October 1984. also: ElectroScience Laboratory Report 714190-9.

[12] R. C. Gonzalez and M. G. Thomason, *Syntactic Pattern Recognition: An Introduction.* Reading, Massachusetts: Addison-Wesley, 1978.

[13] A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translation and Compiling.* Volume 1, Englewood Cliffs: Prentice-Hall, 1972.

[14] K. S. Fu and T. L. Booth, "Grammatical inference: Introduction and survey–Part I," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-5, no. 1, pp. 59-72, January 1975.

[15] A. W. Bierman and J. A. Feldman, "On the synthesis of finite-state acceptors," Artificial Intelligence Memo 114, Computer Science Department, Stanford University, 1970.

END
8-87
DTIC